CONVOLUTION INTERLEAVER AND DEINTERLEAVER FOR SYSTEMS WITH ERROR CORRECTION ENCODING

BACKGROUND

1. Field of the Invention

The present invention relates to telecommunications apparatus, systems and methods. More particularly, the present invention relates to convolutional interleavers and deinterleavers of digital modems and transceivers. The invention has particular application to digital subscriber line (DSL) and wireless systems, although it is not limited thereto.

2. State of the Art

In data communications systems, noisy transmission lines can cause data to be corrupted or lost. In order to prevent data loss it is well known to transmit error-checking data along with the transmitted (payload) data. The combination of the payload data and the error-checking data for the payload is often referred to as a codeword. The receiving end of the data transmission can determine if errors have occurred in a particular codeword based on the payload data and error-checking data received. If the amount of error or loss in a codeword is relatively small, the error-checking data can be used to recover the correct payload

data.

Bursts of noise on the transmission medium may cause the data corruption in a codeword to be sufficiently severe such that recovery is not possible. In order to reduce the impact of these bursts of noise, data interleaving techniques are often employed. By interleaving data from different codewords before data transmission and deinterleaving the received data at the receiver, the impact of a burst of noise is spread over a number of different codewords thereby reducing the loss of each codeword to a level where recovery of the payload data in each codeword is possible.

Well known convolutional interleavers which are used, for example, in ADSL systems, distribute incoming symbols of a codeword according to

$$dL(i) = (D-1)*i, \quad i=0,1,2,...,(N-1) \qquad (1)$$

where dL(i) is the delay of the i'th symbol of a codeword, D is the interleaving depth, and N is the codeword length. Direct implementation of this algorithm in a shift register requires a large memory because the memory size must accommodate the maximum required delay (max dL(i)). The required memory size m_s is defined by:

$$m_s = \max dL(i) = (D-1)*(N-1) \qquad (2)$$

It will be appreciated by those skilled in the art that the memory

size can result in a considerable implementation problem when N and D are large. For example, when using a Reed-Solomon code with codeword length N=255 and interleaving depth D=64, memories of 16,002 8-bit memory cells are required for implementation of the interleaver and deinterleaver. In integrated circuit implementation, large memory blocks require large amounts of "real estate", thereby adding to the cost of the system. Thus, it is desirable to decrease the amount of memory required for implementation of the interleaver and deinterleaver of the system.

In developing interleaving and deinterleaving algorithms which require less memory, it is useful to determine the minimum number of memory cells required for interleaver and deinterleaver implementation. In finding the minimum number of cells required, it may be assumed that every incoming symbol is written into the cell that is released by reading a current outgoing symbol.

Using the first memory cell (MC) to delay (store) the second incoming symbol of the codeword, and adding one more MC every time there is not an empty MC among the MCs which have been already taken, the number of MCs required increases up to a minimum number m_{min}. The minimum m_{min} is the necessary and sufficient number of memory cells required to implement interleaving and is less than m_s.

The number m_{min} is equal to the number of incoming symbols which have come before the m_s'th incoming symbol, but which, on the other hand should be transmitted during or after the m_s'th incoming symbol. In other words, m_{min} is equal to a number of incoming symbols with indexes $n \geq 0$, and $n \leq [(N-1)*(D-1)-1]$, which satisfy the inequality

$$n + (D-1)*r \geq (D-1)*(N-1) \qquad (3)$$

where

$$r = n - (N*q); \quad q = \mathrm{floor}(n/N) \qquad (4)$$

Computer simulation of this algorithm has shown that a number of indexes satisfying inequality (3) is approximately equal to

$$m_{min} \approx (D-1)*(N-1)/2 \qquad (5)$$

Thus, it is seen that the necessary and sufficient number of memory cells for implementing an interleaver is approximately half the number utilized for a standard shift register implementation.

An interleaver/deinterleaver algorithm which can be implemented with a number of memory cells close to m_{min} is disclosed in U.S. Patent #5,636,224 to Voith et al., entitled "Method and Apparatus for Interleave/De-Interleave Addressing in Data Communication Circuits" which is hereby incorporated by reference herein in its entirety. The Voith et al. algorithm

uses parallel circular buffers or FIFOs. The algorithm uses N-1 subsets of memory cells (registers), and symbols of a code word are written into separate registers. In reading symbols from registers of the transmitter and writing symbols to registers of the receiver, the Voith et al. algorithm requires computations which include the solving of some specific equation. These computations must be done "on the fly", and therefore require significant computational power. Thus, the Voith et al. algorithm trades off a decrease in memory requirements for an increase in computational power and as a result does not necessarily reduce the expense of the system significantly.

SUMMARY OF THE INVENTION

It is therefore an object of the invention to provide an interleaver/deinterleaver which can read/write symbols from/to memory registers without current computations.

It is another object of the invention to provide an interleaver/deinterleaver which requires a memory which is not much larger than the minimum required.

It is a further object of the invention to generate a permutation table which is generated based on a simple computation algorithm.

In accord with the objects of the invention, an interleaver for a digital modem is provided where consecutive codeword symbols are written into first cells of parallel registers, and reading from registers for interleaving purposes is determined by a permutation table (register) containing N-1 numbers calculated prior to data transmission. The permutation table contains a sequence of N-1 numbers where each number is an index of a register from which the current symbol should be read.

According to the method of the invention, prior to data transmission, a calculation is made of lengths of the registers (i.e., how many cells each register must have), and a permutation table is determined. Then, during data transmission, the first symbol in each code word is sent directly to the output, the other symbols are sequentially written into the first (input) cells of the corresponding registers, and at the same time, the last (output) cells of the registers are sequentially read in the order determined by the permutation table. After writing the incoming codeword into memory and reading the outgoing codeword, the contents of all memory registers are synchronously shifted by one

cell in the output direction.

A deinterleaver according to the invention corresponds closely to the interleaver of the invention.

Additional objects and advantages of the invention will become apparent to those skilled in the art upon reference to the detailed description taken in conjunction with the provided figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a high-level flow/block diagram of a DSL modem;

Fig. 2 is a high-level block/flow diagram of the interleaving and deinterleaving mechanism of the invention;

Fig. 3 is a chart illustrating an interleaving algorithm example of the invention; and

Fig. 4 is a chart illustration a deinterleaving algorithm example of the invention.

Figs. 5a and 5b are flow charts representing the method of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is best understood with reference to a DSL-type modem. Turning to Figure 1, a high level block diagram of a DSL modem 10 is seen. The modem 10 preferably includes a digital interface 20, a transmitter section 30 and a receiver section 40. The transmitter section preferably includes a scrambler 52 which receives data from the digital interface 20, a Reed-Solomon encoder 54, an interleaver 56, a per-carrier bit distributor 58, a mapper 60, a gain element 62, an inverse fast Fourier transform block (IFFT) 64, a cyclic extension block 66, a digital to analog converter 68 and an analog front end transmit block 69 which interfaces with a hybrid 70. The receiver section preferably includes an analog front end receive block 71 which interfaces with the hybrid 70, an analog to digital converter 72, a time equalizer (TEQ) 73, a fast Fourier transform block (FFT) 74, a frequency equalizer (FEQ) 76, a demapper 78, a deinterleaver 80, a Reed-Solomon decoder 82, and a descrambler 84 which provides data to the digital interface 20. Other than the details of the interleaver 56 and deinterleaver 80, the modem 10 is substantially as would be understood by those skilled in the art. In addition,

it will be appreciated by those skilled in the art that the modem 10 may be implemented in hardware, software, or a combination thereof.

Turning now to Fig. 2, the interleaver 56 and deinterleaver 80 of the invention are seen in high level format. In particular, the interleaver 56 includes a plurality of parallel registers 100 labeled M_1, M_2,... M_{N-2}, M_{N-1}, means 102 for writing symbols into the registers, and means 104 for reading symbols out of the registers 100 according to a permutation table stored in a permutation register 106. The deinterleaver 80 includes a plurality of parallel registers 150 labeled MD_0, MD_1, MD_2,... MD_{N-2}, MD_{N-1}, means 152 for writing symbols into the registers according to the permutation table stored in a permutation register 106, and means 154 for reading symbols out of the registers 150.

The permutation table is generated according to a permutation algorithm which includes, prior to data transmission, calculating the lengths of the registers 100 (and 150) and calculating the permutation table. During data transmission, the inputs and outputs of the registers 100 and 150 are switched according to the permutation table. The permutation table contains a sequence of N-1 numbers (from 1 to N-1), with each number being an index of a register from which the current symbol should be read (in the

interleaver) or an index of a register to which the current symbol should be written (in the deinterleaver).

Fig. 3 illustrates an example of the interleaving algorithm where the codeword length N = 7, and the interleaving depth D = 4. The first three columns in Fig. 3 represent parameters of symbols which are to be interleaved (the "incoming" symbols), with the first column containing the ordinal numbers (indexes) of the symbols, the second column containing the ordinal numbers of the same symbols within a codeword, and the third column representing symbol delays. The last column of Fig. 3 represents indexes of the outgoing symbols; i.e., the interleaved signals which are being currently transmitted. The "memory states" and "permutations" columns of Fig. 3 demonstrate how the incoming symbols are transformed into the outgoing symbols (for transmission). More particularly, when the first codeword of seven symbols is received, the first symbol (byte 0) is used as the first symbol of the interleaved outgoing signal, and the second through seventh symbols (symbols 1-6) are placed into the first cells of registers M_1 through M_6. As will be appreciated by those skilled in the art, because symbol 1 is delayed by only three symbols, it appears as the fifth symbol in the first outgoing word; while, because symbols 2-6 are delayed six, nine, twelve, fifteen, and eighteen symbols respectively, they appear in

the second, third, and fourth outgoing words. Thus, the first outgoing codeword contains incoming symbols 0 and 1, and five additional symbols (bytes) of stuff (denoted by "x").

When the second incoming codeword is received, the first symbol (incoming symbol 7) is used as the first symbol of the interleaved outgoing codeword. The second through seventh symbols (symbols 8-13) are placed into registers M_1 through M_6. Symbol 8 is placed into the first and only cell of register M_1 (which is "empty" because symbol 1 which had previously occupied that cell, was written into the first outgoing codeword). Symbols 9-13 are placed into the first cells of registers M_2 through M_6, the contents of those cells being shifted into the next cells of those registers. The second outgoing codeword is then generated using symbol 7 as the first symbol of that codeword, symbol 2 as the second symbol (symbol 2 having been delayed six symbols), symbol 8 as the fifth symbol (symbol 8 having been delayed three symbols), and symbol 3 as the sixth symbol (symbol 3 having been delayed nine symbols). Three additional symbols of stuff are inserted into the third, fourth, and seventh symbol locations of the second codeword.

When the third incoming codeword is received, the first sybmol (incoming symbol 14) is used as the first symbol of the interleaved outgoing codeword. The second through seventh symbols (symbols 15-20) are placed into registers M_1 through M_6. Symbol 15 is placed into the first and only cell of register M_1 (which is "empty" because symbol 8 which had previously occupied that cell, was written into the second outgoing codeword). Symbols 15-20 are placed into the first cells of registers M_2 through M_6, the contents of those cells being shifted into the next cells of those registers (with the content of the second cells, being shifted into the third cells in the cases of M_4 through M_6). The third outgoing codeword is then generated using symbol 14 as the first symbol of that codeword, symbol 9 as the second symbol (symbol 9 having been delayed six symbols), symbol 4 as the third symbol (symbol 4 having been delayed twelve symbols), symbol 15 as the fifth symbol (symbol 15 having been delayed three symbols), symbol 10 as the sixth symbol (symbol 10 having been delayed nine symbols), and symbol 5 as the seventh symbol (symbol 5 having been delayed fifteen symbols). An additional symbol of stuff is inserted into the fourth byte location of the third codeword.

When the fourth incoming codeword is received, the first symbol (incoming symbol 21) is used as the first symbol of the

interleaved outgoing codeword. The second through seventh symbols (symbols 22-27) are placed into registers M_1 through M_6. Symbol 22 is placed into the first and only cell of register M_1 (which is "empty" because symbol 15 which had previously occupied that cell, was written into the third outgoing codeword). Symbols 22-27 are placed into the first cells of registers M_2 through M_6, the contents of those cells being shifted into the next cells of those registers (with the content of the second cells, in the case of M_4 and M_5, being shifted into the third cells, and the content of the third cell, in the case of M_6, being shifted into a fourth cell). The fourth outgoing codeword is then generated using symbol 21 as the first symbol of that codeword, symbol 16 as the second symbol (symbol 16 having been delayed six symbols), symbol 11 as the third symbol (symbol 11 having been delayed twelve symbols), symbol 6 as the fourth symbol (symbol 6 having been delayed eighteen symbols), symbol 22 as the fifth symbol (symbol 22 having been delayed three symbols), symbol 17 as the sixth symbol (symbol 17 having been delayed nine symbols), and symbol 12 as the seventh symbol (symbol 12 having been delayed fifteen symbols).

When the fifth incoming codeword is received, the process is repeated as indicated. No additional cells are required in

registers M_1 through M_6 as the contents contained in one of the cells of each of the registers was written into the previous outgoing codeword. Thus, it should be appreciated that in the case of the interleaving algorithm where N=7, and D=4, six registers are required, with the first register containing only a single cell, the second and third registers containing two cells, the fourth and fifth registers containing three cells, and the sixth register containing four cells. The number of cells in each of the registers is calculated in advance and is not changed during data transmission. In addition, the order from which the last of the cells of each of the registers is sequentially read is repetitive (as seen by comparing the groups of arrows of the permutation column of Fig. 3 with respect to each codeword) and therefore may be predicted according to a permutation table:

Table 1: Interleaver permutation table for N=7, D=4

Byte of outgoing codeword	0	1	2	3	4	5	6
Register byte is taken from	Taken directly	2	4	6	1	3	5

Thus, the first symbol of every outgoing codeword is taken from the first incoming symbol directly; the second symbol of the outgoing codeword is taken from the second cell of the second register M_2 (having been delayed six symbols); the third symbol of the outgoing codeword is taken from the third cell of the fourth register M_4 (having been delayed twelve symbols); the fourth symbol of the outgoing codeword is taken from the fourth cell of the sixth register M_6 (having been delayed eighteen symbols); the fifth symbol of the outgoing codeword is taken from the first and only cell of the first register M_1 (having been delayed three symbols); the sixth symbol of the outgoing codeword is taken from the second cell of the third register M_3 (having been delayed nine symbols); and the seventh symbol of the outgoing codeword is taken from the third cell of the fifth register M_5 (having been delayed fifteen symbols).

Once the interleaver permutation table is set, the deinterleaver permutation table is effectively determined. In particular, and as seen in Fig. 4, the symbols of the incoming codeword on the deinterleaving side are distributed to seven parallel registers in an inverse manner according to the following deinterleaver table.

Table 2: Deinterleaver permutation table for N=7, D=4

Byte of interleaved incoming codeword	0	1	2	3	4	5	6
Register to which byte is directed	0	2	4	6	1	3	5

Thus, symbol 0 is forwarded to register MD_0; symbol 1 is forwarded to register MD_2; symbol 2 is forwarded to register MD_4; symbol 3 is forwarded to register MD_6; symbol 4 is forwarded to register MD_1; symbol 5 is forwarded to register MD_3; and symbol 6 is forwarded to register MD_5. The registers MD_0 through MD_6 are likewise provided with different numbers of cells depending upon how long bytes must be stored before an entire codeword can be reconstructed. As seen in the memory states column of Fig. 4, the number of cells for registers MD_1 through MD_6 is essentially the reverse of the number of cells for registers M_1 through M_6 of the interleaver, and the extra register (M_0) is provided with four cells.

With the register structure provided, and utilizing the deinterleaving table, the codewords are regenerated by taking the symbols located in the last cells of each of the registers in order. As shown in Fig. 4, with N=7 and D=4, the first regenerated symbol occurs after receiving the fourth incoming codeword. Thereafter, all cells of all seven registers are utilized with symbols being written into the first cells of each of the registers, and symbols already sitting in cells either being shifted into adjacent cells or being written out to regenerate a codeword.

Because the interleaver and deinterleaver use a simple table, complex computations are not required to implement them. In addition, the interleaver and deinterleaver of the invention do not require memories which are significantly larger than the minimum required. In fact, in the general case (for odd N, and any value for D), the total number of interleaver memory cells m_I is equal to

$$m_I = (N-1)*[((D-1)/2) + 1)] \qquad (6)$$

which is close to the minimum m_{min} and almost two time less than m_s. If N=255 and D=64, then m_I = 8255, m_{min} = 8001, and m_s = 16002. Similarly, the total number of interleaver memory cells

$m_D = m_I + D$. Thus, for the same example where N=255 and D=64, m_D = 8319.

Given all of the above, methods of interleaving and deinterleaving according to the invention can be described according to initialization steps and processing steps. The steps utilize the following designations:

D - interleaving depth;

N - code word length;

M_j - j'th interleaver register;

L_j - j'th interleaver register length;

MD_j - j'th deinterleaver register;

LD_j - j'th deinterleaver register length;

P - permutation register with length N-1;

$S_{in}(i)$ - i'th incoming symbol of a codeword;

ceil - rounding up to the next whole number;

floor - rounding down to the next whole number.

Referring now to Fig. 5a, according to a method of the invention, in a first initialization step for interleaving, the length L_j of the j'th register M_j is determined at 200 according to:

$$L_j = \mathrm{ceil}(j*D/N),\ j=1,2,\ldots,(N-1) \tag{7}$$

In a second initialization step for interleaving, at 205, a permutation register holding indications of the interleaving permutation table is initialized. Initialization can be represented by calculating

$$f(n) = n_{ModN} \tag{8}$$

$$k(n) = (D-1)*f(n) + n \tag{9}$$

$$m(n) = k(n) - N*(D-1) \tag{10}$$

where $n=0,1,2,\ldots,(N*D - 1)$,

and by setting indication f(n) into the m(n)'th cell of the permutation register P when $N*D>k(n)>N*(D-1)$.

Initialization may also be represented by the following code:

```
for n=0:1:(N*D - 1)
    f=n-N*floor(n/N);
    k=(D-1)*f + n;
    if k>N*(D-1) and k<N*D
        m=k-N*(D-1);
        P(m) = f;
    end
end
```

The results of the above code for N=7 and D=4 are illustrated in Appendix 1 below:

Appendix 1

n	f=n-7*floor(n/7)	k=n+3*f	If k>21 and k<28 m= k-21	P(m)=f
0	0	0	-	-
1	1	4	-	-
2	2	8	-	-
3	3	12	-	-
4	4	16	-	-
5	5	20	-	-
6	6	24	3	P(3)=6
7	0	7	-	-
8	1	11	-	-
9	2	15	-	-
10	3	19	-	-
11	4	23	2	P(2)=4
12	5	27	6	P(6)=5
13	6	31	-	-
14	0	14	-	-
15	1	18	-	-
16	2	22	1	P(1)=2
17	3	26	5	P(5)=3
18	4	30	-	-
19	5	34	-	-
20	6	38	-	-
21	0	21	-	-
22	1	25	4	P(4)=1
23	2	29	-	-
24	3	33	-	-
25	4	37	-	-
26	5	41	-	-
27	6	45	-	-

In a third initialization step for interleaving, at 210, N-1 registers M_j are initialized with lengths L_j. Memory set M is an association of M_j such that $M = [M_1, M_2 \ldots M_{N-1}]$.

After initialization of the registers M_j and the permutation register P, codewords having incoming symbols $S_{in}(i)$ (with indexes i=0,1,2,...N-1 where i=0 corresponds to the first element of a codeword) are processed (interleaved) as follows. First, a determination is made at 215 whether or not i=0. If i=0, at 220, the current outgoing symbol is set equal to a current incoming symbol; i.e.,

$$S_{out} = S_{in}(0) \tag{11}$$

Then, at 225 the contents of all registers are shifted by one cell towards the register output; i.e.,

$$M_j(m_j) = M_j(m_j-1),\ m_j = 2,3,\ldots,L_j;\ j=1,2,\ldots,N-1 \tag{12}$$

On the other hand, if at 215 i>0, the current incoming symbol $S_{in}(i)$ is written at 230 in the now vacant first memory cell of the i'th register M_i; i.e.,

$$M_i(1) = S_{in}(i) \tag{13}$$

and at 235 a current outgoing symbol is read from the shifted out last memory cell of the P(i)'th register $M_{P(i)}$; i.e.,

$$S_{out} = M_{P(i)}(L_{P(i)}) \quad (14)$$

After step 225 or step 235, the index i is incremented at 240, and a check is made at 245 as to whether i=N. If i=N, at 248, i is reset to zero, and the loop continues at step 215. If i≠N, i is not reset, but the loop is continued at step 215.

The initialization steps and processing steps for deinterleaving correspond closely to the initialization and processing steps discussed above with respect to interleaving. As seen in Fig. 5b, a first initialization step 250 for deinterleaving involves calculating register lengths LD_j which represent the number of memory cells in each of the registers MD_j according to:

$$LD_j = D - \mathrm{floor}(j*D/N), \quad j=0,1,2,\ldots,(N-1) \quad (15)$$

It should be noted that when parameters N and D are equal in both the upstream and downstream directions, LD_j may be also found by reordering the L_j according to

$$LD_0 = D \quad (16)$$

$$LDj = L_{N-j} \quad j=1,2,\ldots,(N-1) \quad (17)$$

A second initialization step 255 for deinterleaving involves initializing the permutation register P(j) which is identical to

the interleaving initialization of the permutation register, except that P(0) is set equal to 0, such that the first incoming symbol of the incoming interleaved codeword is placed into the first register of the deinterleaving registers.

A third initialization step 260 for deinterleaving involves initializing N registers MD_j with lengths LD_j. The full memory set MD is an association of MD_j:

$$MD = [MD_0, MD_1 \ldots MD_{N-1}] \tag{18}$$

After initialization of the registers MD_j and the permutation register P, codewords having incoming symbols $S_{in}(i)$ (with indexes i=0,1,2,...N-1 where i=0 corresponds to the first element of a codeword) are processed (deinterleaved) as follows. First, at 265, the current incoming symbol $S_{in}(i)$ is written in the first memory cell of the P(i)'th register $MD_{P(i)}$ according to:

$$MD_{Pi}(1) = S_{in}(i) \tag{20}$$

Then, at 270, a current outgoing symbol S_{out} is read from the last memory cell of the i'th register MD_i; i.e.,

$$S_{out} = MD_i(L_{P(i)}) \tag{21}$$

At 275, i is incremented, and at 280 a determination is made as to

whether i = N. If i = N, at 285 the contents of all registers are shifted by one cell towards the register output; i.e.,

$$MD_j(m_j) = MD_j(m_j-1),\ m_j = 2,3,\ldots,L_j;\ j=0,1,2,\ldots,N-1 \qquad (22)$$

and, at 290, i is reset to zero. The loop then continues at 265. If at 280, on the other hand, i ≠ N, the loop continues at 265 with symbols being written into memory cells.

A MATLAB® simulation program for the interleaver initialization and processing and the deinterleaver initialization and processing is attached as Appendix 2 below:

Appendix 2 - Interleaver/Deinterleaver Simulation in Matlab

```
N=input('enter Code Length N ');
D=input('enter Interleaving Depth D ');
%message generation
M=D+5;                                  %example: for algorithm checking
Bin=[0 1:1:(M*N-1)];                    %example: Incoming Data
%INTERLEAVER
%Registers Size Calculation
Sz=zeros(size(1:(N-1)));
for qq=1:(N-1)
   Sz(qq)=ceil(qq*D/N);
end
Sz;                               %demo:Interleaver Registers Size
MemoryI=sum(Sz);                  %demo:Number of Interleaver Memory Cells
%Registers initialization
R=zeros((N-1),D);            %In real implementation: Ri=zeros(size(1:(Sz(i))))
%Calculation of the Permutation Table
P=zeros(size(1:N));
for q=1:N*D
   qm=q-1;
   ff=floor(qm/N); f=qm-N*ff;
   qk=(D-1)*f+qm;
   if  qk>N*(D-1) & qk<N*D
      m=qk-N*(D-1);
      P(m)=f;
   end
end
P';                               %demo: The Permutation Table
```

1

```
%Interleaver Processing
for n=1:(N*M)
   nn=n-1;ii=floor(nn/N);i=nn-N*ii;
   if i==0
      Bout=Bin(n);
      z=2:1:D; R(:, z)=R(:,z-1);
   else
      R(i,1)=Bin(n);
      Bout=R(P(i),Sz(P(i)));
   end
   XByout(n)=Bout;              %Interleaver Output
   DBin(n)=Bout;                %Deinterleaver Input
end                             %end of interleaver processing
XByout';                        %demo:Interleaver Outgoing Data
%DEINTERLEAVER
%Register Sizes Calculation
Sz=[D zeros(size(1:(N-1)))];
for qq=1:(N-1)
   Sz(N+1-qq)=ceil(qq*D/N);
end
Sz;                       %demo:Registers Size
MemoryD=sum(Sz);          %demo:Number of Deinterleaver Memory Cells
%Registers initialization
R=zeros(N,D);          %In real implementation: Ri=zeros(size(1:(Sz(i))))
P=P+1;
%Deinterleaver Processing
for n=1:(N*M)
   nn=n-1;ii=floor(nn/N);i=nn-N*ii;j=i+1;
   if i==0
      R(1,1)=DBin(n);
   else
      R(P(i),1)=DBin(n);
   end
   DBout=R(j,Sz(j));
   if j==N
      z=2:1:D; R(:,z)=R(:,z-1);
   end
   DByout(n)=DBout;             %Deinterleaver Output
end
DByout'                         %demo:Deinterleaver Output
MemoryI
MemoryD
```

There has been described and illustrated herein a preferred embodiment of an interleaver and a deinterleaver for a digital modem. While a preferred embodiment of the invention has been described, it is not intended that the invention be limited thereto, as it is intended that the invention be as broad in scope as the art will allow and that the specification be read likewise. Thus, while the invention was described with reference to symbols which are each a byte long, it will be appreciated that the symbols could be of other lengths. Also, while particular code has been listed for initializing the permutation register, it will be appreciated that other code could be utilized. Likewise, while particular Matlab code has been provided for implementing interleaver and deinterleaver initialization and processing, other code could be utilized. Further, while the invention has been described with reference to "registers", it will be appreciated by those skilled in the art that the term "register" is intended to be broadly understood to include all different types of storage elements including FIFOs, shift-registers, circular buffers, RAM, etc. Indeed, it should be noted that where the invention is implemented with a well-known circular buffer rather than with shift registers, a physical shift of data (as suggested in Figs. 5a and 5b) is not required. Instead, the equivalent is accomplished through the adjustment of read and write pointers. It will also be appreciated that the registers, read means, and

write means of the invention may be implemented in hardware, software, or a combination thereof. In addition, while the permutation register of the invention has been described as holding N-1 cells, it will be appreciated that the permutation register can include an additional cell (cell 0) which is set to value 0. Also, while the size of each cell of the permutation registers was not specified, it will be appreciated that the permutation register cells can be byte-wide for convenience, or may be sized simply to accommodate the number of symbols in the codewords; i.e., if the codeword has seven bytes, only three bits are needed in each permutation register cell to identify each register, whereas if the codeword has 128 bytes, seven bits would be needed.

It will further be appreciated by those skilled in the art that while the invention was described as using the permutation register for reading data out of the interleaver, in fact, by rearranging the cell depths of the respective registers, the permutation register may be utilized for writing data into the registers. Thus, the first incoming symbol (symbol 0) will continue to be directly read out. Using the permutation register, the second incoming symbol is then written to the fourth register M_4 (which is provided with only a single cell), the third incoming symbol is written to the first register M_1 (which is provided with

two cells), the fourth incoming symbol is written to register M_5 (which is provided with two cells), the fifth incoming symbol is written to M_2 (which is provided with three cells), the sixth incoming symbol is written to M_6 (which is provided with three cells), and the seventh incoming symbol is written to M_3 (which is provided with four cells). Reading of the symbols out of the registers may then be conducted in direct order; i.e., after reading out symbol 0 directly, registers M_1 through M_6 are read in sequential order. In a similar fashion, instead of using the permutation register to write data into the deinterleaver, the permutation register may be used to read data out of the deinterleaver. Where the permutation register is used in that manner, interleaved symbols may be read in sequential order into the deinterleaver registers which are provided with appropriate depths.

It should be appreciated that the contents of the permutation register used to write data into the interleaver (instead of reading data out of the interleaver) may be derived from the contents of the permutation register which reads data out of the interleaver. This is simply accomplished by causing the register contents (i.e., register # for reading data) to be used as the register address (i.e., outgoing codeword symbol #), and vice

versa. Thus, for example, in Table 1, where the contents of symbol 1 is taken from register 2 (for a permutation register used to read data out of the interleaver), symbol 2 would be taken from register 1 (for the permutation register used to write data into the interleaver). Likewise, in Table 1, where the contents of symbol 2 are taken from register 4, in the permutation register used to write data into the interleaver, the contents of symbol 4 are taken from register 2. As a result, Table 1 can be extended as follows:

Outgoing codeword symbol #	0	1	2	3	4	5	6
Register # for reading data	Taken directily	2	4	6	1	3	5
Register # for writing data	Taken directly	4	1	5	2	6	3

and the permutation register can be taken from either row 2 or row 3 of the above table depending upon whether the permutation register is being used to read data out of registers or write data into registers. It should be noted that Table 2 may also be similarly extended.

Those skilled in the art will also appreciate that while Figs. 5a and 5b suggest that for every symbol written into the interleaver or deinterleaver, another symbol is read out, the order may be different. For example, an entire codeword may be written in and then an entire codeword read out.

It will therefore be appreciated by those skilled in the art that yet other modifications could be made to the provided invention without deviating from its spirit and scope as so claimed.